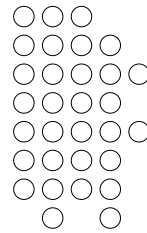


# 第七章 迴圈

結構化程式設計  
for、while 與 do while 迴圈  
選擇適當的迴圈敘述  
如何跳離迴圈



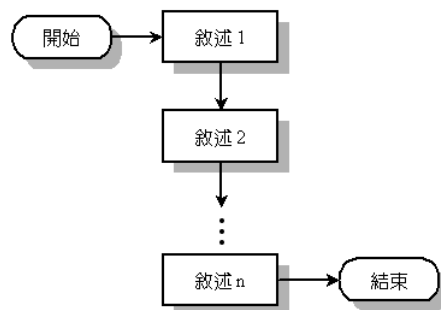
## 結構化程式設計

- 結構化的程式設計包含有下面三種結構：
  - 循序性結構 (sequence structure)
  - 選擇性結構 (selection structure)
  - 重復性結構 (repetition structure)



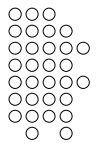
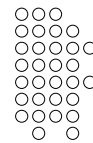
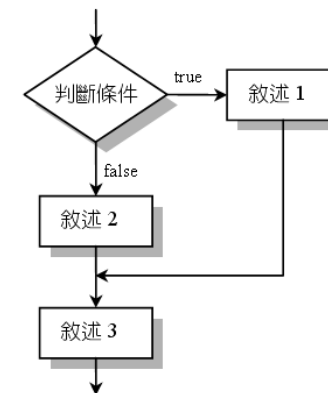
## 結構化程式設計

- 循序性結構 (sequence structure)
  - 程式的執行流程是由上而下，一個接著一個敘述依序執行



## 結構化程式設計

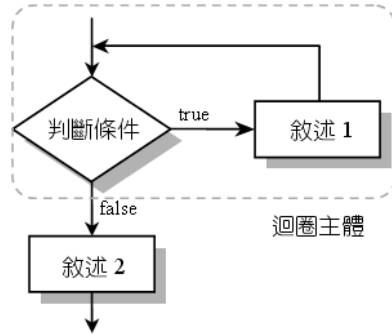
- 選擇性結構 (selection structure)
  - 依判斷條件的結果來決定程式執行的順序



# 結構化程式設計

- 重複性結構 (iteration structure / repetition structure)

- 在符合判斷條件時反覆執行迴圈主體 (敘述 1)
- 直到不符合判斷條件才離開，繼續向下執行 (敘述 2)



# 使用 for 迴圈

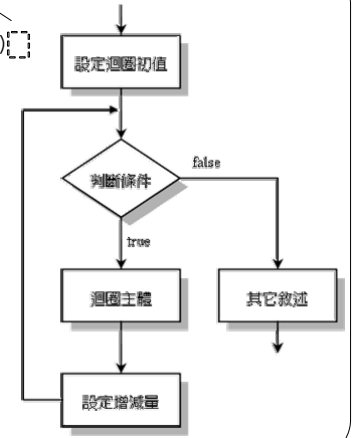
這兒不可以加分號

for 迴圈的語法

```
for (設定迴圈初值; 判斷條件; 設定增減量) {
    迴圈主體;
}
```

這兒不需要加分號

```
for ( i=1, sum=0; i<=9; i+=2 ) {
    迴圈主體;
}
```

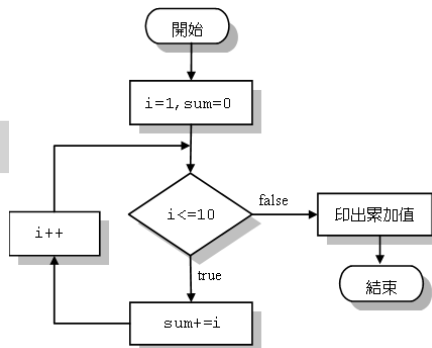


# for 迴圈範例一 (1/2)

- 利用 for 迴圈計算 1 加到 10 的總和

```
01 /* prog7_1, for 迴圈的使用 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06     int i, sum=0;
07     for(i=1; i<=10; i++)
08         sum+=i;
09     printf("1+2+3+...+10=%d\n", sum);
10
11     system("pause");
12     return 0;
13 }
```

/\* prog7\_1 OUTPUT--  
1+2+3+...+10=55  
-----\*/



迴圈控制變數的命名常用 i, j, k, ...

# for 迴圈範例一 (1/2)

- 迴圈內，變數變化的情形

```
06 int i, sum=0;
07 for(i=1; i<=10; i++)
08     sum+=i;
```

表 7.2.1 for 迴圈內，i 與 sum 值變化的情形

i 的值	sum 的值	計算 sum+=i 之後，sum 的值
1	0	1
2	1	3
3	3	6
4	6	10
5	10	15
6	15	21
7	21	28
8	28	36
9	36	45
10	45	55

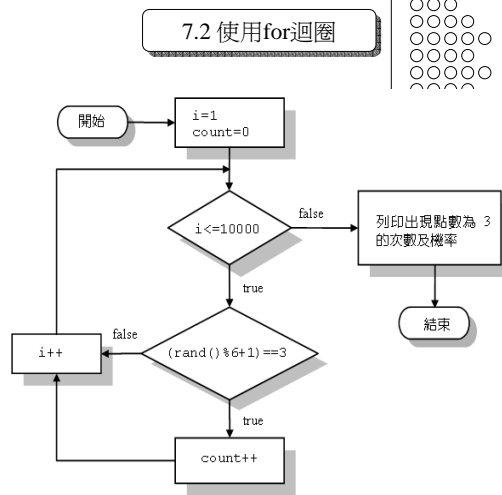
執行完 for 迴圈之後，sum 的值

## for 迴圈範例二

- 隨機實驗的模擬與機率的計算

```

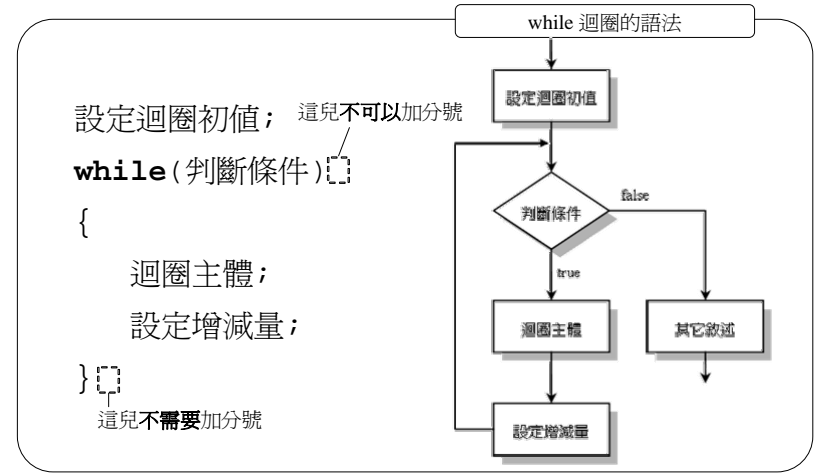
01  /* prog7_2, 使用 for 迴圈計算機
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      int i, count=0;
07
08      for(i=1; i<=10000; i++)
09          if ((rand()%6+1)==3)
10              count++;
11      printf("擲 10000 次骰子時, 出現 3 點的次數為 %d 次\n", count);
12      printf("機率為 %.3f\n", (float) count/10000);
13
14      system("pause");
15      return 0;
16  }
    
```



## 7.3 使用while迴圈

## 使用 while 迴圈

- while 最適合用在迴圈執行次數為未知時



## while 迴圈的範例

- 將 while 用在未知執行次數的迴圈

```

01  /* prog7_3, while 迴圈的使用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      int i=1, sum=0; /* 設定迴圈初值 */
07      while(sum<=100) /* while 迴圈, 當 sum 小於 100 則繼續累加 */
08      {
09          sum+=i;
10          printf("從 1 累加到 %2d=%2d\n", i, sum);
11          i++;
12      }
13      printf("必須累加到 %d\n", i-1);
14      system("pause");
15      return 0;
16  }
    
```

/\* prog7\_3 OUTPUT ---  
 從 1 累加到 1= 1  
 從 1 累加到 2= 3  
 ...  
 從 1 累加到 14=105  
 必須累加到 14  
 -----\*/

## 7.3 使用while迴圈

## for 與 while 迴圈的比較

- 理論上 for 迴圈與 while 迴圈是等效的敘述

表 7.3.1 for 迴圈與 while 迴圈的敘述比較

for 迴圈	while 迴圈
<pre> for (設定初值; 判斷條件; 設定增減量) {     敘述 1;     敘述 2;     ...     敘述 n; }                     </pre>	<pre> 設定初值; while (判斷條件) {     敘述 1;     敘述 2;     ...     敘述 n;     設定增減量; }                     </pre>

## 無窮迴圈的範例一

- 當迴圈判斷條件一直滿足時，即稱為無窮迴圈

```

01 /* prog7_4, 無窮迴圈的說明 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06     int i=1;
07
08     while (i > 0) /* 當 i>0 時執行 while 迴圈的主體 */
09         printf("i=%d\n",i++);
10
11     system("pause"); /* prog7_4 OUTPUT---
12     return 0;
13 }

```

這個迴圈是會停下來  
while (i>0);  
或是  
while (i>0)  
printf("i=%d\n", i);  
才是永遠不會停的

```

i=1
i=2
i=3
... (無窮迴圈的輸出)
-----*/

```

13

## 無窮迴圈的範例二

- 利用無窮迴圈持續輸入字元：

```

01 /* prog7_5, 無窮迴圈的應用 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06     char ch;
07     while(ch!=17) /* 當按下的鍵不是 Ctrl+q 時 */
08     {
09         ch=getch(); /* 從鍵盤取得字元 */
10         printf("ASCII of ch=%d\n",ch); /* 印出取得字元的 ASCII 碼 */
11     }
12     printf("您已按了 Ctrl+q...\n");
13
14     system("pause");
15     return 0;
16 }

```

沒有初始化

```

/* prog7_5 OUTPUT---
ASCII of ch=17
ASCII of ch=104
ASCII of ch=13
ASCII of ch=17
您已按了 Ctrl+q...
-----*/

```

```

double x;
for (x=0.;x!=100.;x+=0.1)
while (1)
for (x=0.;x!=100.;x+=0.1)

```

14

## 使用 do while 迴圈

設定迴圈初值；

do

{

迴圈主體；

設定增減量；

} while(判斷條件);

需加分號

注意：判斷條件為迴圈繼續執行的條件，不要想成是結束的條件

迴圈主體至少會執行過一次

do-while 迴圈的語法



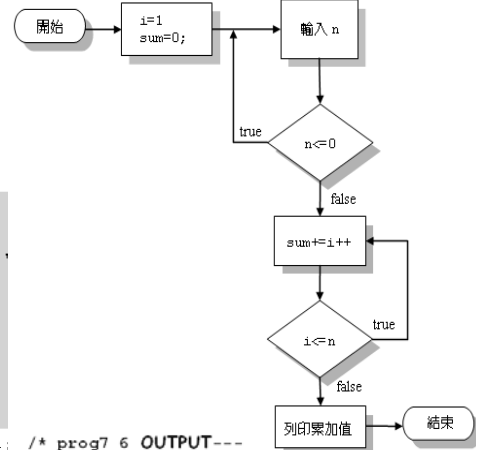
15

## do while 迴圈的範例一

```

01 /* prog7_6, do while 迴圈 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06     int n,i=1,sum=0;
07     do
08     {
09         printf("請輸入 n 值 (n>0): ");
10         scanf("%d",&n);
11     }
12     while (n<=0);
13     do
14     {
15         sum+=i++;
16         while (i <= n);
17         printf("1+2+...+%d=%d\n",n,sum);
18     }
19 }

```



```

/* prog7_6 OUTPUT---
請輸入 n 值 (n>0): -6
請輸入 n 值 (n>0): 10
1+2+...+10=55
-----*/

```

16



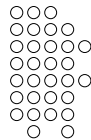
# 空迴圈

- 迴圈主體內沒有任何的敘述，稱為空迴圈

### 空迴圈的說明

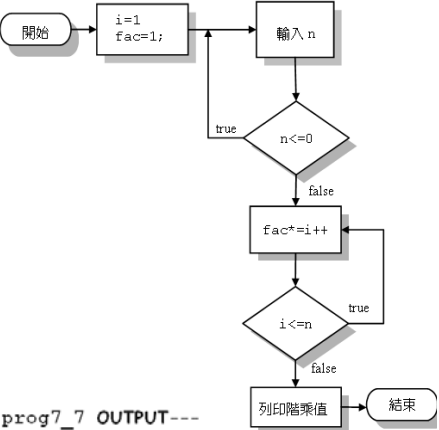
```
for(設定初值;判斷條件;設定增減量)
{
}
或是
for(設定初值;判斷條件;設定增減量){}
```

這兒要加分號



# do while 迴圈的範例二

```
01 /* prog7_7, 利用 do while 迴圈求 n!
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06     int n,i=1,fact=1;
07     do
08     {
09         printf("請輸入 n 值 (n>0): ");
10         scanf("%d",&n);
11     }
12     while (n<=0);
13     do
14         fact*=i++;
15     while (i <= n);
16     printf("%d!=%d\n",n,fact); /* prog7_7 OUTPUT---
17     system("pause");
18     return 0;
19 }
```



```
請輸入 n 值 (n>0): -3
請輸入 n 值 (n>0): 6
6!=720
```



# 使用哪一種迴圈？

- 下表列出了每一種迴圈的特性比較：

表 7.6.1 for、while 與 do while 迴圈的比較

迴圈特性	迴圈種類		
	for	while	do while
前端測試判斷條件	是	是	否
後端測試判斷條件	否	否	是
於迴圈主體中需要更改控制變數的值	否	是	是
迴圈控制變數會自動變更	否	否	否
迴圈重複的次數	已知	未知	未知
至少執行迴圈主體的次數	0 次	0 次	1 次
何時重複執行迴圈	條件成立	條件成立	條件成立

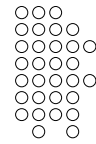


# 空迴圈的範例

- 下面範例的空迴圈常常是寫程式時不小心留下的 bug

```
01 /* prog7_8, 空迴圈的誤用 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06     int i;
07     for(i=1;i<=10000;i++); /* 空迴圈 */
08     printf("i=%d\n",i);
09
10     system("pause");
11     return 0;
12 }
```

```
/* prog7_8 OUTPUT--
i=10001
-----*/
```



## 巢狀迴圈的範例-九九乘法表(1/2)

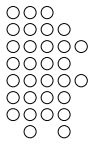
- 迴圈裡又有另一層迴圈，稱為巢狀迴圈
- 九九乘法表的列印：

```

/* prog7_9 OUTPUT-----
1*1= 1 1*2= 2 ... 1*9= 9
2*1= 2 2*2= 4 ... 2*9=18
...
9*1= 9 9*2=18 ... 9*9=81
-----*/
01 /* prog7_9, 巢狀 for 迴圈印出九九乘法表 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06     int i,j;
07     for (i=1;i<=9;i++) /* 外層迴圈 */
08     {
09         for (j=1;j<=9;j++) /* 內層迴圈 */
10             printf("%d*%d=%2d ",i,j,i*j); } 內層迴圈 } 外層迴圈
11     printf("\n");
12 }
13 system("pause");
14 return 0;
15 }

```

21



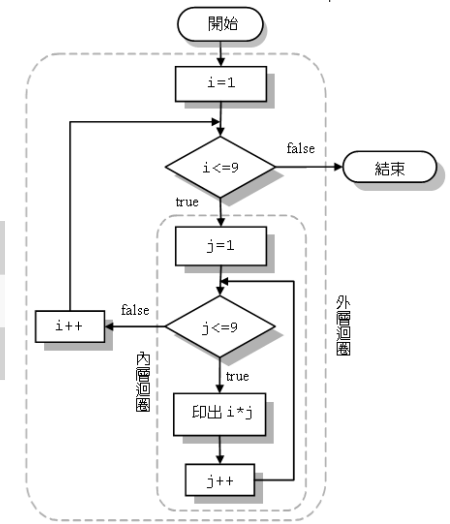
## 巢狀迴圈的範例-九九乘法表(2/2)

- 九九乘法表的流程圖：

```

for (i=1;i<=9;i++) /* 外層迴圈 */
{
    for (j=1;j<=9;j++) /* 內層迴圈 */
        printf("%d*%d=%2d ",i,j,i*j);
    printf("\n");
}

```



## 以巢狀 while 迴圈改寫九九乘法表

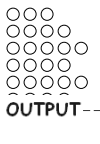
```

01 /* prog7_10, 巢狀 while 迴圈求 9*9 乘法表 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06     int i=1, j=1; /* 設定迴圈控制變數的初值 */
07     while (i<=9) /* 外層迴圈 */
08     {
09         while (j<=9) /* 內層迴圈 */
10         {
11             printf("%d*%d=%2d ",i,j,i*j);
12             j++;
13         }
14         printf("\n");
15         i++;
16         j=1;
17     }
18     system("pause");
19     return 0;
20 }

```

• 本範例的流程圖與執行結果同 prog7\_9

23



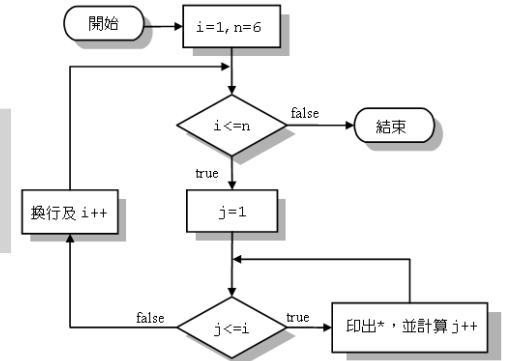
## 以巢狀 for 迴圈印出幾何圖形

- 利用巢狀迴圈印出三角形

```

/* prog7_11 OUTPUT--
*
**
***
****
*****
-----*/
01 /* prog7_11, 利用巢狀迴圈印出三角形 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06     int i,j,n=6;
07
08     for (i=1;i<=n;i++)
09     {
10         for (j=1;j<=i;j++)
11             printf("*");
12         printf("\n");
13     }
14
15     system("pause");
16     return 0;
17 }

```



24

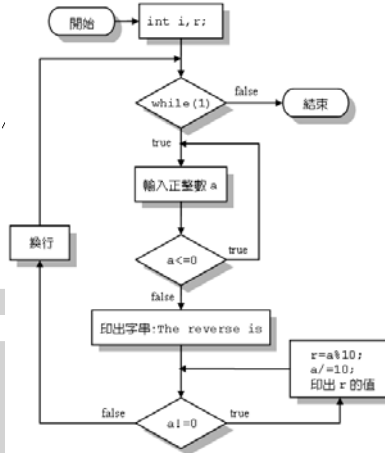
## 以巢狀迴圈反印數字(1/2)

- 將整數反印，例如 5123→3215

```

01 /* prog7_12, 巢狀迴圈, 將整數反過來列印 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06     int a, r;
07
08     while(1)
09     {
10         do
11         {
12             printf("Input an integer:");
13             scanf("%d",&a);
14         }
15         while (a<=0); /* 必須輸入大於 0 的正整數 */
16

```



25

## 以巢狀迴圈反印數字(2/2)

```

17     printf("The reverse is ");
18     while (a!=0)
19     {
20         r=a%10; /* 計算 a/10 的餘數 */
21         a/=10; /* 計算 a/10, 再設回給 a */
22         printf("%d", r);
23     }
24     printf("\n\n");
25 }
26 system("pause");
27 return 0;
28 }

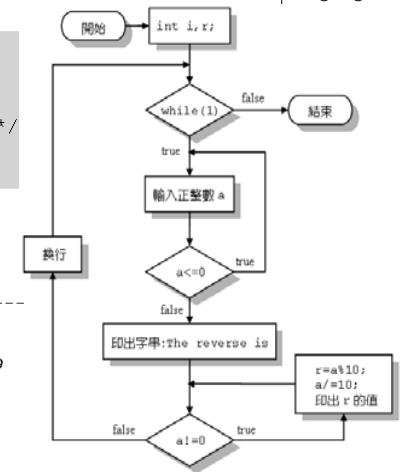
```

/\* prog7\_12 OUTPUT----

Input an integer:-58  
Input an integer:13579  
The reverse is 97531

Input an integer:2468  
The reverse is 8642

Input an integer:  
-----\*/



26

## 迴圈的強制跳離

- break** 敘述：
  - 略過迴圈主體的其餘部分，結束迴圈，執行迴圈之後的敘述

## break 敘述的語法

```
for(初值設定; 判斷條件; 設定增減量)
```

```
{
    敘述 1;
    敘述 2;
    ...

```

```
break;
```

```
...
    敘述 n;
}
```

若執行break敘述，則此區塊內的敘述不會被執行

27

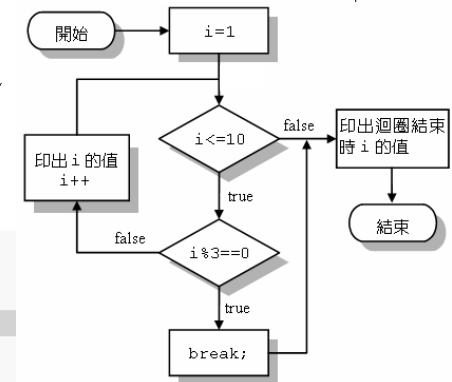
## break 敘述

- break** 敘述的範例：

```

01 /* prog7_13, break 敘述的使用 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06     int i;
07     for(i=1; i<=10; i++)
08     {
09         if(i%3==0)
10             break; /* 跳離迴圈 */
11         printf("i=%d\n", i);
12     }
13     printf("跳離迴圈時, i=%d\n", i);
14
15     system("pause");
16     return 0;
17 }

```



/\* prog7\_13 OUTPUT--

i=1  
i=2  
跳離迴圈時, i=3  
-----\*/

28



## continue 敘述

### • continue 敘述：

- 略過迴圈主體的其餘部分，直接開始下一個迴圈循環 (由判斷條件開始測試)

#### continue 敘述的語法

```
for (初值設定; 判斷條件; 設定增減量)
```

```
{
  敘述 1;
  敘述 2;
```

```
...
  continue;
```

```
...
  敘述 n; } 若執行continue敘述，則此
  ...      區塊內的敘述不會被執行
```

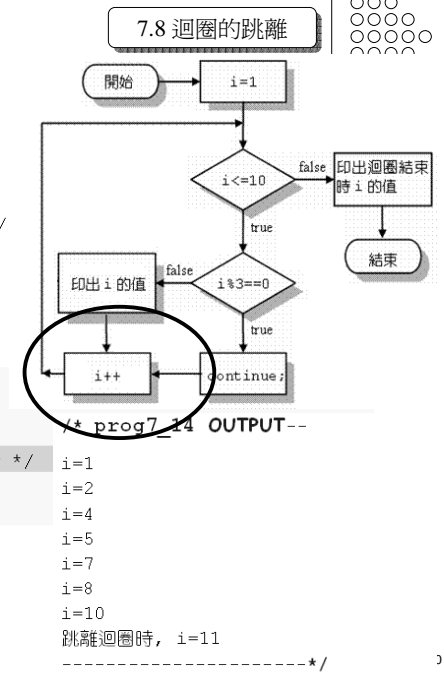


29

## continue 敘述

### • continue 敘述的範例

```
01 /* prog7_14, continue 敘述的使用 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06     int i;
07     for(i=1;i<=10;i++)
08     {
09         if(i%3==0)
10             continue; /* 回到起始處執行 */
11         printf("i=%d\n",i);
12     }
13     printf("跳離迴圈時, i=%d\n",i);
14     system("pause");
15     return 0;
16 }
17
```



```
/* prog7_14 OUTPUT--
i=1
i=2
i=4
i=5
i=7
i=8
i=10
跳離迴圈時, i=11
-----*/
```

30

## 如何提早結束多層迴圈



```
for (i=0; i<10; i++) {
    for (j=0; j<20; j++) {
        if (count<0) break;
        ... // count 變數內容會改變
    }
    if (j<20) break; // 或是 if (count<20) break;
    ... // count 變數內容會改變
}
// 正常結束的話 i==10&&j==20
```

方法一

變化很多，請靈活運用

```
for (i=0; i<10; i++) {
    for (j=0; j<20; j++) {
        if (count<0) {j=19; continue;}
        ... // count 變數內容會改變
    }
    if (count<0) {i=9; continue;}
    ... // count 變數內容會改變
}
// 正常與否 i==10&&j==20
```

方法二

```
for (i=0, stop=0; i<10&&!stop; i++) {
    for (j=0; j<20&&!stop; j++) {
        if (count<0) {stop=1; continue;}
        ... // count 變數內容會改變
    }
    if (stop) continue;
    ... // count 變數內容會改變
}
// 正常結束的話 i==10&&j==20
```

方法三

31